

Docket No.: J0658.0014  
(PATENT)

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

---

In re Patent Application of:  
Robert E. Ober et al.

Application No.: 10/712,473

Confirmation No.: 9335

Filed: November 12, 2003

Art Unit: 2183

For: INTERRUPT AND TRAP HANDLING IN  
AN EMBEDDED MULTI-THREAD  
PROCESSOR TO AVOID PRIORITY  
INVERSION AND MAINTAIN REAL-TIME  
OPERATION

---

Examiner: B. P. Johnson

**APPEAL BRIEF**

MS Appeal Brief - Patents  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

Dear Sir:

As required under § 41.37(a), this brief is filed within two months of the Notice of Appeal filed in this case on January 24, 2007, and is in furtherance of said Notice of Appeal.

The fees required under § 41.20(b)(2) are submitted herewith.

This brief contains items under the following headings as required by 37 C.F.R. § 41.37 and M.P.E.P. § 1206:

I.	Real Party In Interest
II	Related Appeals and Interferences
III.	Status of Claims
IV.	Status of Amendments
V.	Summary of Claimed Subject Matter
VI.	Grounds of Rejection to be Reviewed on Appeal
VII.	Argument
VIII.	Claims
IX.	Evidence
X.	Related Proceedings
XI.	Claims Appendix
XII.	Evidence Appendix
XIII.	Related Proceedings Appendix

I. REAL PARTY IN INTEREST

Based on information supplied by Appellant and to the best of the Appellant's legal representative's knowledge, the real party of interest is the assignee, Infineon Technologies AG.

II. RELATED APPEALS, INTERFERENCES, AND JUDICIAL PROCEEDINGS

There are no other appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Pursuant to the final Office Action dated July 25, 2006, claims 1-5, 9-16, 20-22, 25, 26, 29-32 and 34-35 remain rejected under 35 § U.S.C. 102(b) over Hobbs (US Patent No. 5,197,138), and claims 6-8, 17-19, 27-28, 33 and 36 remain rejected under 35 U.S.C. § 103(a) over Hobbs in view of Radhakrishna (U.S. Patent No. 6,823,414). Claims 23 and

24 are canceled. Thus, claims 1-22 and 25-36 are pending in the application, with all pending claims on appeal.

#### IV. STATUS OF AMENDMENTS

Applicant filed an Amendment After Final Rejection on October 25, 2006; claim 24 was canceled. The Amendment was entered, as indicated in the November 13, 2006 Advisory Action.

#### V. SUMMARY OF CLAIMED SUBJECT MATTER

*Paragraph numbers in this section correspond with those of the published application.*

Independent claim 1 is directed to a method for operating a multi-threaded system having a plurality of active threads. [See paragraph 26 and Abstract.] In the method, an interrupt priority value is assigned to each of a plurality of interrupts. [See paragraph 26.] An interrupt threshold value is specified. [See paragraph 27 and Abstract.] A requested interrupt is processed only when the interrupt priority value of the requested interrupt is higher than the interrupt threshold value. [See paragraphs 27, 28, and 31, Abstract, and Fig. 4A.]

Independent claim 22 is directed to a method for operating a multi-threaded system having a plurality of active threads. [See paragraph 26 and Abstract.] In the method, a request for an interrupt is accepted. [See paragraphs 26, 28, and 31, and Fig. 4A.] Execution is switched to a predetermined one of the plurality of active threads. [See paragraph 26.] An interrupt service request from the predetermined one of the plurality of active threads is executed to service the interrupt. [See paragraphs 26, 28, and 31.] Accepting a request for an interrupt includes assigning a unique interrupt priority value to each interrupt for the multi-threaded embedded system [see paragraph 26],

specifying a common threshold interrupt value for all the active threads [*see paragraph 27 and Abstract*], and taking the interrupt only when the unique interrupt priority value of the interrupt is higher than the common threshold interrupt value [*see paragraphs 27, 28, and 31, Abstract, and Fig. 4A*].

Independent claim 25 is directed to a multi-threaded system. [*See paragraph 26 and Abstract.*] The system includes a thread execution logic for generating instruction requests from an executing thread. [*See paragraph 26.*] A threshold interrupt logic for generating an interrupt threshold value, wherein the thread execution logic only accepts interrupts having an interrupt priority higher than the interrupt threshold value. [*See paragraphs 27, 28, and 31, Abstract, and Fig. 4A.*]

Independent claim 29 is directed to a multi-threaded system. [*See paragraph 26.*] The system includes a means for specifying an interrupt threshold value. [*See paragraph 27 and Abstract.*] The system also includes a means for processing a requested interrupt only when an interrupt priority value of the requested interrupt is higher than the interrupt threshold value. [*See paragraphs 27, 28, and 31, Abstract, and Fig. 4A.*]

## VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. Whether claims 1-5, 9-16, 20-22, 25, 26, 29-32 and 34-35 were erroneously rejected under 35 § U.S.C. 102(b) over Hobbs (US Patent No. 5,197,138).

B. Whether claims 6-8, 17-19, 27-28, 33 and 36 were erroneously rejected under 35 U.S.C. § 103(a) over Hobbs in view of Radhakrishna (U.S. Patent No. 6,823,414).

## VII. ARGUMENT

### A. Independent Claim 1, and Dependent Claims 2-21

Independent claim 1 recites "processing a requested interrupt only when the interrupt priority value of the requested interrupt is higher than the interrupt threshold value." The benefits of the claimed interrupt threshold value are explained in paragraph 27 of the published application, where it discusses that the fixed interrupt threshold value (ITV) provides a global interrupt priority value for all active threads. Since any interrupt accepted by thread execution logic must have a higher interrupt priority value than this global interrupt threshold value, any interrupt being serviced will always have a higher interrupt priority than any of the active threads, thereby preventing priority inversion.

In contrast, Hobbs discloses that "[a]n interrupt will not be recognized or serviced by the processor until the priority of the code thread is lower than the priority of the interrupt." (See col. 2, lines 54-57.) Thus, Hobbs compares the priorities of two different processor functions (i.e., code thread and interrupt), wherein each of these priorities varies. This is different from determining whether a priority of an interrupt is higher than a fixed threshold value, as claimed. Therefore, independent claim 1, along with its dependent claims 2-5, 9-16, 20, and 21, is patentable.

Radhakrishna was applied as allegedly teaching features recited in dependent claims 6-8 and 17-19. Radhakrishna fails to make up for the deficiencies of Hobbs. Thus, claims 6-8 and 17-19 are patentable by virtue of their dependence on independent claim 1.

B. Independent Claim 22

Independent claim 22 recites "specifying a common threshold interrupt value for all the active threads," and "taking the interrupt only when the unique interrupt priority value of the interrupt is higher than the common threshold interrupt value." The benefits of the claimed interrupt threshold value are explained in paragraph 27 of the published application, where it discusses that the fixed interrupt threshold value (ITV) provides a global interrupt priority value for all active threads. Since any interrupt accepted by thread execution logic must have a higher interrupt priority value than this global interrupt threshold value, any interrupt being serviced will always have a higher interrupt priority than any of the active threads, thereby preventing priority inversion.

In contrast, Hobbs discloses that "[a]n interrupt will not be recognized or serviced by the processor until the priority of the code thread is lower than the priority of the interrupt." (See col. 2, lines 54-57.) Thus, Hobbs compares the priorities of two different processor functions (i.e., code thread and interrupt), wherein each of these priorities varies. This is different from taking an interrupt only when a unique interrupt priority value of the interrupt is higher than a specified common threshold interrupt value, as claimed. Therefore, independent claim 22 is patentable.

C. Independent Claim 25, and Dependent Claims 26-28

Independent claim 25 recites "threshold interrupt logic for generating an interrupt threshold value, wherein the thread execution logic only accepts interrupts having an interrupt priority higher than the interrupt threshold value." The benefits of the claimed interrupt threshold value are explained in paragraph 27 of the published application, where it discusses that the fixed interrupt threshold value (ITV) provides a

global interrupt priority value for all active threads. Since any interrupt accepted by thread execution logic must have a higher interrupt priority value than this global interrupt threshold value, any interrupt being serviced will always have a higher interrupt priority than any of the active threads, thereby preventing priority inversion.

In contrast, Hobbs discloses that "[a]n interrupt will not be recognized or serviced by the processor until the priority of the code thread is lower than the priority of the interrupt." (See col. 2, lines 54-57.) Thus, Hobbs compares the priorities of two different processor functions (i.e., code thread and interrupt), wherein each of these priorities varies. This is different from a thread execution logic only accepting interrupts having an interrupt priority higher than an interrupt threshold value, as claimed. Therefore, independent claim 25, along with its dependent claim 26, is patentable.

Radhakrishna was applied as allegedly teaching features recited in dependent claims 27 and 28. Radhakrishna fails to make up for the deficiencies of Hobbs. Thus, claims 27 and 28 are patentable by virtue of their dependence on independent claim 25.

#### D. Independent Claim 29, and Dependent Claims 30-36

Independent claim 29 recites a "means for specifying an interrupt threshold value," and a "means for processing a requested interrupt only when an interrupt priority value of the requested interrupt is higher than the interrupt threshold value." The benefits of the claimed interrupt threshold value are explained in paragraph 27 of the published application, where it discusses that the fixed interrupt threshold value (ITV) provides a global interrupt priority value for all active threads. Since any interrupt accepted by thread execution logic must have a higher interrupt priority value than this global interrupt threshold value, any interrupt being serviced will always

have a higher interrupt priority than any of the active threads, thereby preventing priority inversion.

In contrast, Hobbs discloses that "[a]n interrupt will not be recognized or serviced by the processor until the priority of the code thread is lower than the priority of the interrupt." (See col. 2, lines 54-57.) Thus, Hobbs compares the priorities of two different processor functions (i.e., code thread and interrupt), wherein each of these priorities varies. This is different from a means for processing a requested interrupt only when an interrupt priority value of the requested interrupt is higher than a specified interrupt threshold value, as claimed. Therefore, independent claim 29, along with its dependent claims 30-32, 34, and 35, is patentable.

Radhakrishna was applied as allegedly teaching features recited in dependent claims 33 and 36. Radhakrishna fails to make up for the deficiencies of Hobbs. Thus, claims 33 and 36 are patentable by virtue of their dependence on independent claim 29.

### VIII. CLAIMS

A copy of the claims involved in the present appeal is attached hereto as Claims Appendix.

### IX. EVIDENCE

No evidence pursuant to §§ 1.130, 1.131, or 1.132 or entered by or relied upon by the examiner is being submitted.

### X. RELATED PROCEEDINGS

No related proceedings are referenced in II. above.

Please charge any fee, except for the Issue Fee, that may be necessary for the continued pendency of this application to our Deposit Account No. 04-0100.

Dated: February 26, 2007

Respectfully submitted,

By   
Laura C. Brutman

Registration No.: 38,395  
DICKSTEIN SHAPIRO LLP  
1177 Avenue of the Americas  
41st Floor  
New York, New York 10036-2714  
(212) 277-6500  
Attorney for Applicant

**CLAIMS APPENDIX**

**CLAIMS 1-22 AND 25-36 ARE ON APPEAL:**

1. A method for operating a multi-threaded system having a plurality of active threads, the method comprising:

assigning an interrupt priority value to each of a plurality of interrupts;

specifying an interrupt threshold value; and

processing a requested interrupt only when the interrupt priority value of the requested interrupt is higher than the interrupt threshold value.

2. The method of claim 1, wherein processing the requested interrupt comprises:

performing an interrupt entry process to prepare for an interrupt service routine (ISR);

executing the ISR; and

performing an interrupt exit process to return control from the ISR.

3. The method of claim 2, wherein performing the interrupt entry process comprises:

identifying one of the plurality of active threads as an interrupt thread;

switching to the interrupt thread if the interrupt thread is not executing; and  
branching to the ISR.

4. The method of claim 3, wherein each of the plurality of active threads comprises a thread context, and wherein performing the interrupt entry process further comprises saving the thread context of the interrupt thread.

5. The method of claim 4, wherein performing the interrupt exit process comprises:

executing a return from exception (RFE) instruction; and  
restoring the thread context of the interrupt thread.

6. The method of claim 5, wherein performing the interrupt entry process further comprises disabling interrupts and thread switching.

7. The method of claim 6, wherein performing the interrupt exit process further comprises enabling interrupts and thread switching.

8. The method of claim 6, wherein executing the ISR comprises enabling interrupts and thread switching after a predetermined interval.

9. The method of claim 3, wherein each of the plurality of active threads consists of a first set of context registers and a second set of context registers, wherein performing the interrupt entry process further comprises saving the first set of context registers of the interrupt thread, and wherein executing the ISR comprises:

saving the second set of context registers of the interrupt thread if the second set of context registers of the interrupt thread are required for servicing the requested interrupt;

servicing the requested interrupt; and

restoring the second set of context registers of the interrupt thread after servicing the requested interrupt if the second set of context registers of the interrupt thread were required for servicing the requested interrupt.

10. The method of claim 9, wherein performing the interrupt exit process comprises:

executing a return from exception (RFE) instruction; and

restoring the upper context registers of the interrupt thread.

11. The method of claim 1, further comprising processing traps only in the active threads originating the traps.

12. The method of claim 11, wherein processing traps comprises:

detecting a trap from an originating thread, the originating thread being one of the plurality of active threads;

storing trap background data for the trap if the trap is asynchronous; and

associating a trap pending indicator with the originating thread if the originating thread is not executing.

13. The method of claim 12, wherein processing traps further comprises:

performing a trap entry process to prepare for a trap handling routine;

executing the trap handling routine; and

performing a trap exit process to return control from the trap handling routine.

14. The method of claim 13, wherein each of the plurality of active threads comprises a thread context, and wherein performing the trap entry process comprises:

waiting for originating thread to start executing if the originating thread is not executing;

saving the thread context of the originating thread; and branching to a trap handler.

15. The method of claim 14, wherein waiting for the originating thread to start executing comprises monitoring the plurality of active threads until execution switches to one of the plurality of active threads associated with the trap pending indicator.

16. The method of claim 14, wherein performing the trap exit process comprises:

executing a return from trap instruction; and

restoring the context of the originating thread.

17. The method of claim 16, wherein performing the trap entry process further comprises disabling interrupts and thread switching.

18. The method of claim 17, wherein performing the trap exit process further comprises enabling interrupts and thread switching.

19. The method of claim 17, wherein executing the trap handling routine comprises enabling interrupts and thread switching after a predetermined interval.

20. The method of claim 13, wherein each of the plurality of active threads consists of a first set of context registers and a second set of context registers, wherein

performing the trap entry process further comprises saving the first set of context registers of the originating thread, and wherein executing the trap handling routine comprises:

saving the second set of context registers of the originating thread if the second set of context registers of the originating thread are required for servicing the trap;

servicing the trap; and

restoring the second set of context registers of the originating thread after servicing the trap if the second set of context registers of the interrupt thread were required for servicing the trap.

21. The method of claim 20, wherein performing the interrupt exit process comprises:

executing a return from trap instruction; and

restoring the upper context registers of the originating thread.

22. A method for operating a multi-threaded system having a plurality of active threads, the method comprising:

accepting a request for an interrupt;

switching execution to a predetermined one of the plurality of active threads;  
and

executing an interrupt service request from the predetermined one of the plurality of active threads to service the interrupt,

wherein accepting a request for an interrupt comprises:

assigning a unique interrupt priority value to each interrupt for the multi-threaded embedded system;

specifying a common threshold interrupt value for all the active threads; and

taking the interrupt only when the unique interrupt priority value of the interrupt is higher than the common threshold interrupt value.

25. A multi-threaded system comprising:

thread execution logic for generating instruction requests from an executing thread; and

threshold interrupt logic for generating an interrupt threshold value, wherein the thread execution logic only accepts interrupts having an interrupt priority higher than the interrupt threshold value.

26. The multi-threaded system of claim 25, further comprising interrupt thread logic for switching execution to a selected interrupt thread before servicing any interrupt.

27. The multi-threaded system of claim 26, further comprising disabling logic for disabling interrupts and thread switching while an interrupt is being serviced.

28. The multi-threaded system of claim 27, further comprising thread tagging logic for storing trap background data for asynchronous traps, wherein every trap is handled in its originating thread.

29. A multi-threaded system comprising:

means for specifying an interrupt threshold value; and

means for processing a requested interrupt only when an interrupt priority value of the requested interrupt is higher than the interrupt threshold value.

30. The multi-threaded system of claim 29, wherein the means for processing the requested interrupt comprises:

means for identifying one of the plurality of active threads as an interrupt thread;

means for switching to the interrupt thread if the interrupt thread is not executing; and

means for branching to the ISR.

31. The multi-threaded system of claim 30, wherein the means for processing the requested interrupt further comprises means for saving a thread context of the interrupt thread.

32. The multi-threaded system of claim 31, wherein the means for processing the requested interrupt further comprises means for restoring the thread context of the interrupt thread after executing a return from exception (RFE) instruction.

33. The multi-threaded system of claim 32, wherein the means for processing the requested interrupt further comprises means for disabling interrupts and thread switching during execution of the ISR.

34. The multi-threaded system of claim 29, further comprising means for processing traps, the means for processing traps comprising:

means for detecting a trap from an originating thread;

means for storing trap background data for the trap if the trap is asynchronous;

means for processing the trap if the originating thread is executing; and

means for associating a trap pending indicator with the originating thread if the originating thread is not executing.

35. The multi-threaded system of claim 34, wherein the means for processing traps further comprises means for detecting execution of an active thread associated with the trap pending indicator.

36. The multi-threaded system of claim 35, wherein the means for processing the trap comprises means for disabling interrupts and thread switching.

**EVIDENCE APPENDIX**

All evidence is in the record.

**RELATED PROCEEDINGS APPENDIX**

There are no related proceedings for this matter.